Future Computing and Informatics Journal

Volume 2
Issue 1 (2020) *2017, Volume 2, Issue 1*

Article 1

2017

A supporting tool for requirements change management in distributed agile development

ramdan mowad
Faculty of Computers and Information Technology, Future University in Egypt, Cairo, Egypt, ramdan.mowad@fue.edu.eg

Domia Lloyd
College of Computing and Information Technology, Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt, domia4@yahoo.com

Mona Kadry
College of Computing and Information Technology, Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt, monakadry@yahoo.com

Available online at www.sciencedirect.com

ScienceDirect

Future Computing and Informatics Journal 2 (2017) 1–9
http://www.journals.elsevier.com/future-computing-and-informatics-journal/

A supporting tool for requirements change management in distributed agile development

Domia Lloyd [a], Ramadan Moawad [b,*], Mona Kadry [a]

[a] College of Computing and Information Technology, Arab Academy for Science and Technology and Maritime Transport, Cairo, Egypt
[b] Faculty of Computers and Information Technology, Future University in Egypt, Cairo, Egypt

Received 18 January 2017; revised 23 March 2017; accepted 12 April 2017
Available online 13 May 2017

Abstract

Software development industry has witnessed the growth of the agile movement and approaches. Applying the agile approaches and practices in the distributed environment will lead to gain a lot of benefits such as reduced costs, higher efficiency and better customization, on the other hand it will face many challenges for example working in different time zones, requirements changes, personal selection and knowledge management. In order to gain these benefits, it should address the challenges that will face the agile approaches in a distributed environment. One of the main challenges is managing the requirements changes during the process of distributed agile software development. Only few researches published in the literature, addressed the problem of requirements changes during the development process in distributed agile development. Most of the published researches in this context are based on industrial experiences which increases the need for combining the industry with academia within this area. In this paper an approach to manage requirements changes in distributed agile development is introduced. This suggested approach works to fill the gap between the industry and research in distributed agile development by combining the industrial practice and academic technique. This approach is based on a proposed feature model called a features tree. The approach is associated with a supporting software tool that helps to manage the requirement changes in distributed agile development. The supporting tool is tested and evaluated in real environments by software development professionals using an exhaustive set of criteria, and the results are promising.

Keywords: Distributed agile; Requirements changes

1. Introduction

One of the main challenges that face the process of software requirements is the communication between the onshore and offshore sites in distributed teams. To reduce iteration time at the offshore site requires more efficient communication with the onshore site. Previous knowledge in certain domain is very important to understand software requirements.

Agile methods are considered to be a good solution when they work in small iterations to deliver complete software solution at the end of these iterations. The Agile development aims to increase the team effectiveness, by reducing time of exchanging and sharing information between people [1]. Another aspect of Agile is minimizing the delivered documentation.

In distributed agile, the development activities are divided into two categories: onshore activities and offshore activities [2]. One of the main reasons why the software projects are deployed at offshore is that by taking advantage of cheaper labor, facilities and talented workforce [2]. Reducing the development cost is one of the main advantages that can be achieved using offshore development [3]. Distributing the

* Corresponding author.
E-mail addresses: domia4@yahoo.com (D. Lloyd), ramdan.mowad@fue.edu.eg (R. Moawad), monakadry@yahoo.com (M. Kadry).
Peer review under responsibility of Faculty of Computers and Information Technology, Future University in Egypt.

http://dx.doi.org/10.1016/j.fcij.2017.04.001

development processes between offshore and onshore sites as in distributed agile development will issue the communication challenge. The communication challenge between customers and onshore team from one side and the development team at the offshore on the second side arises from the difficulty to settle face to face meetings and discuss different project issues according to the difference in physical location and hence different geographical areas and time zones. The main challenges in distributed agile are communication, knowledge management, cultural diversity, time differences and changes of requirements [4].

This paper introduces an approach for distributed agile development to manage requirements and their changes during the development processes. This approach works to fill the gap between the industry and research in distributed agile development by combining the industrial practice and academic techniques. The suggested approach is based on a feature model [5] using a feature tree. The feature model is originally introduced to model the commonality and variability in domain engineering phase of software product line development A supporting tool is developed to help managing software requirements changes. Its main objective is to mitigate some of the challenges facing distributed agile development. The supporting tool is tested and evaluated in real environments.

The rest of this paper is organized as the following, Section 2 introduces the basic concepts of the feature model and the authors' extensions to it. Section 3 illustrates the construction process for the features tree. Section 4 presents how to use features tree in distributed agile. The supporting tool is introduced in Section 5. Section 6 presents the evaluation of the supporting tool.

Section 7 presents conclusion and future work.

2. Feature model

Feature modeling is an approach to model software requirements with height complexity and expressing several requirements in features and structure them hierarchically in feature diagrams. The development of software families or product line requires a comprehensive understanding of the domains. A detailed and thorough analysis of the domains must be performed and the results have to be documented in a consistent form. There are number of analysis methodologies exists in the literature, such as Organization Domain Modeling [4], Feature-Oriented Domain Analysis [6], and Domain-Specific Software Architectures [7] are the most popular ones.

Fig. 1 represents an example of the feature model for a simple web registration system. The web registration system consists of three main features: Two mandatory features which are registration and login features, and one optional feature which is the authentication feature. The registration feature contains two mandatory features name and birth date and also contains two alternative features username and mobile number, the last feature is email and it is an optional feature. The second main feature is the login feature which contains one mandatory feature password and two alternative features username and mobile number features.

The selection of Username feature in Login feature requires the selection of Username feature in Registration main feature, as well as the selection of Email feature requires the selection of Email feature in Registration main feature. The third and last main feature is authentication feature and it is an optional feature. Authentication feature contains two alternative features: Email Authentication and SMS authentication. The selection of the SMS feature requires the selection of Mobile No. feature and the selection of Email in Authentication requires the selection of Email in Registration feature. The feature model of web registration system can satisfy the requirements for more than one application in the domain of web registration and that what is called the molding requirements for a family of systems or product line.

The feature model is used to model variability for a family of systems or product line in a specific domain. In our case we won't do that. The aim of our approach is to model the requirements for single software application in distributed agile environment. The software requirements in this case are subject to change from time to time during the development process. We aim to adopt the feature model to manage the requirements' changes and represent these changes as variability to study the impact of these changes to the other requirements which is called requirements dependency. On the other hand modeling requirements changes will help the offshore development team to understand the new requirements and the possibility of development. In the next section we will introduce our suggested feature tree which contains some modifications to the feature model to support the process of managing requirements changes in the distributed agile development.

The adoption process of feature model will include new notations. All requirements will be modeled as a mandatory feature with considerable concentration on the relation between requirements which is called requirements dependency in the form of Implication and Exclusion relationship. There is no need to indicate the optional, mandatory or alternative requirements which is used in product line. To manipulate and manage requirements' changes the authors suggest new notation to represent the processes of adding, updating and deleting requirements. Table 1 presents the suggested notations to maintain and control the requirements' changes processes which will be represented in the feature tree in next section.

3. Features tree

The process of constructing the feature tree consists of two main steps. The first step aims to construct the initial form of the feature tree. The initial form of features tree will contain all candidate requirements. Fig. 2 shows an example of the initial feature tree.

The second step is managing the requirements changes. After each iteration the development team will take the decision of approving the new changes or preventing it using the approval and prevention notations which presented in Table 1. The change management process can be summarized in three activities. a) Modification of requirements in the form of

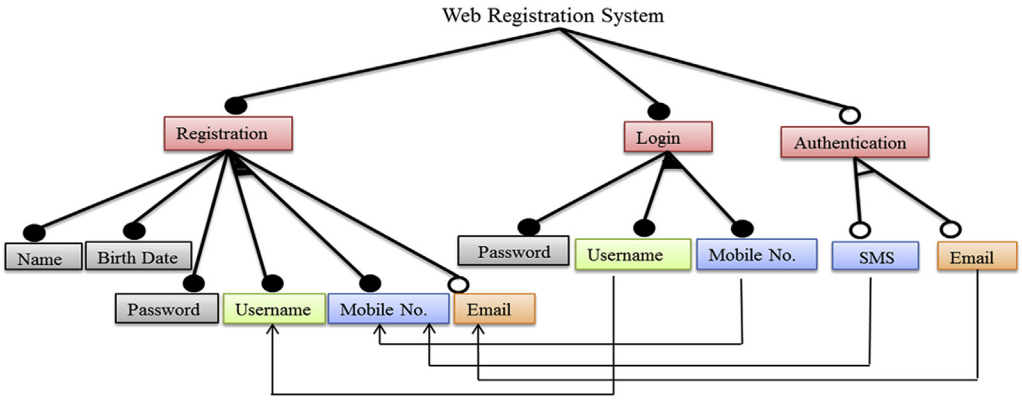
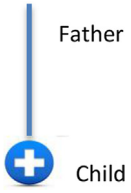
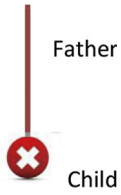
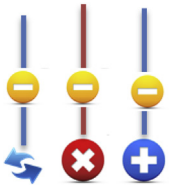


Fig. 1. Web registration feature model.

Table 1
Extended feature model notations.

Type	Semantic	Notation	Type	Semantic	Notation
Add	Add new child feature.		Delete	Delete existing child feature.	
Update	Update existing child feature.		Approve modification	Approve deleting, adding and updating.	
Prevent modification	Prevent deleting, adding and updating		Exclusion	Indicates that both features cannot be selected in one product configuration and are therefore mutually exclusive	
Implication	If one feature is selected the implied feature has to be selected as well, ignoring their position in the feature tree.				

adding, deleting, and updating. b) Controlling of changes in the form of approve or prevent. c) Keep tacking of dependency relationship between requirements in the form of implication and exclusion.

4. Features tree in distributed agile

To introduce a complete distributed agile approach with requirement change management, the authors combined the suggested feature tree with The New Agile Process for Distributed Projects (NAPDP) [8,9]. NAPDP is an approach to handle Global Agile Development (GAD); it provides benefits to the domain knowledge exchange and proposes a better branching and release management. It also provides a set of best practices, helping to link design, test approaches and tools, which help to maintain a software architecture.

NAPDP is based on exploring different approaches which follow the agile development models and the agile manifesto.

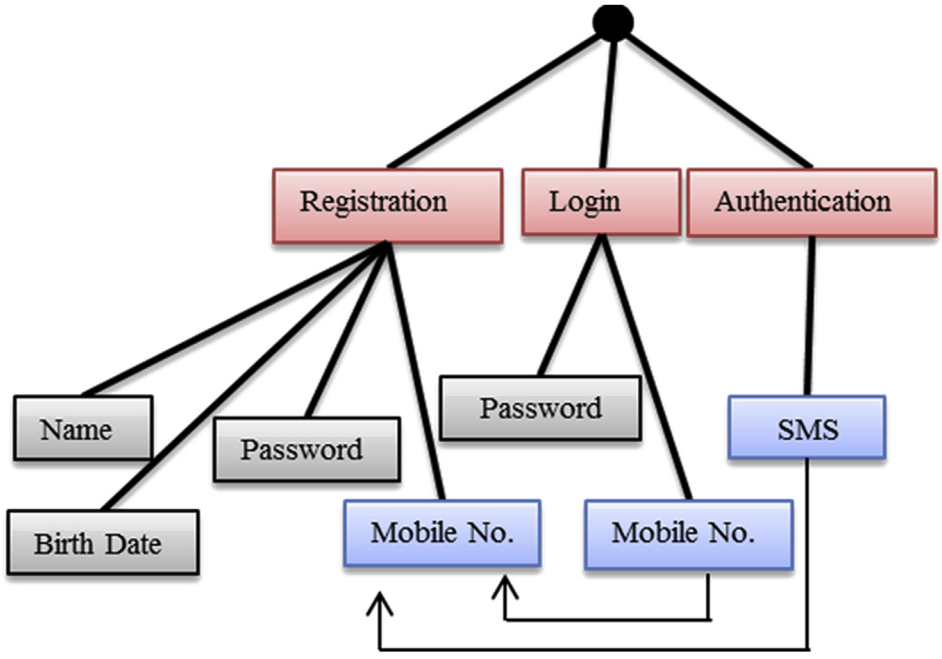


Fig. 2. Example of the feature tree.

NAPDP has combined the most appropriate methods and best practices from Rational Unified Process (RUP) [12,13], Extreme Programming (XP) [10], Scrum and Rapid Object Oriented Process for Embedded System (ROPES) [11]. The suggested approach has taken the advantages of NAPDP and combined these advantages with the suggested feature tree. The proposed approach consists of two main phases: Project Initialization and Development Cycle, each phase contains several steps.

Project initialization phase

During the project initialization phase the project manager and requirements engineers capture requirements from the customer/stakeholders. Many planning activities are also conducted in this phase. This phase consists of the following three steps:

1) Project planning

At the earlier project meetings, several activities are conducted: project planning and scheduling, allocation of resources, development strategies including development plan and configuration management.

2) Requirements Engineering

In this step requirement engineers capture requirements from different resources such as domain experts and stakeholders. The requirements engineers are also responsible for eliciting, analyzing, specifying, validating requirements and building the initial features tree.

3) Creating the Prioritized Feature Lists

In this step the Prioritized Feature Lists are created. Each list will be implemented in a development cycle iteration and integrated into the system. Project manager, software architect, test and quality manager, feature designer, requirements engineers, configuration manager and executive programmers, all are involved in this step.

Development cycle phase

In development cycle phase a certain number of features are implemented. During each development cycle iteration the detailed feature tree is maintained. Features are designed, implemented, and tested. A quick prototype of the implemented features is delivered for customer evaluation at the end of each iteration. In the following section we will explain each step in more details.

1) Analysis

Each development cycle iteration starts with analysis step. This step includes requirement analysis, architecture analysis and object analysis.

2) Maintaining Feature Tree

The process of maintaining feature tree includes adding new features, modifying or updating existing features and deleting existing features. The maintenance process includes also the approval of the requirement change and the tracking of dependency relationship between requirements.

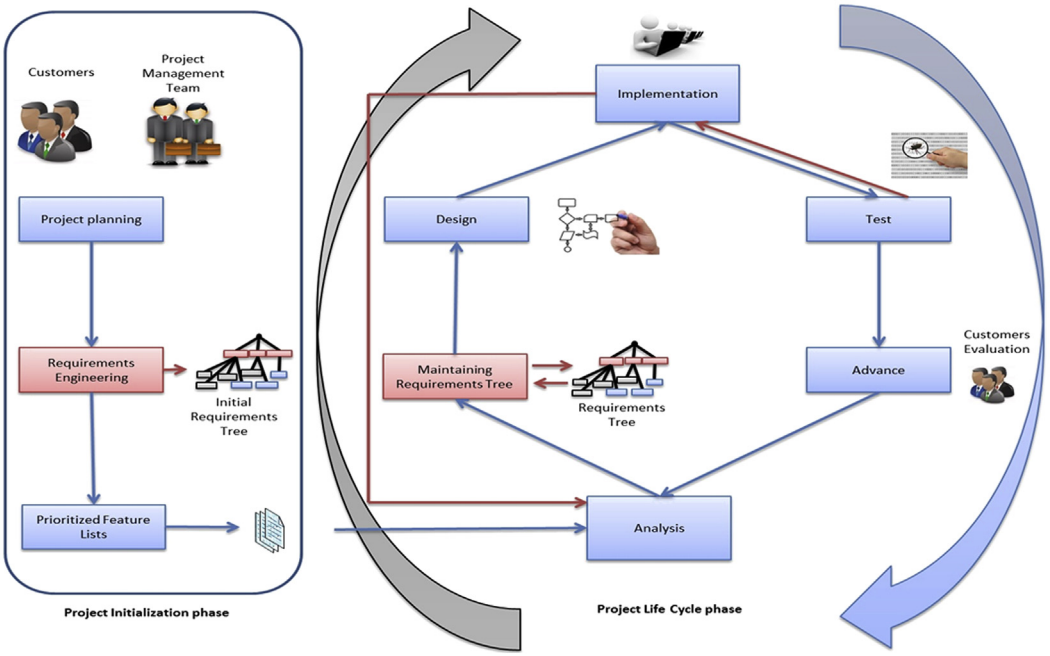


Fig. 3. The phases of suggested approach.

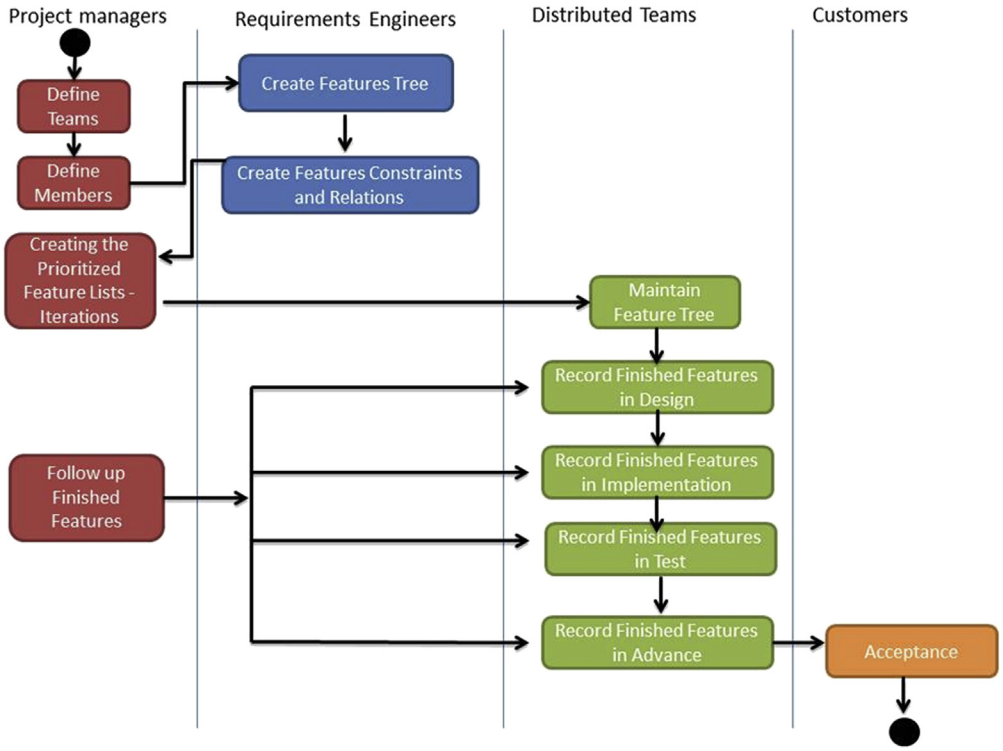


Fig. 4. The activity diagram of supporting tool.

3) Design

After the analysis is completed and the new requirements changes are maintained in the feature tree, the process goes to the design step. This step starts from architectural design.

4) Implementation

The implementation step translates features into achievable solution. The developers write the required code to develop the features.

Fig. 5. Maintaining feature tree in supporting tool.

Table 2
Evaluation criteria.

No.	Criteria	Description
1	Concept	Does the tool provide mechanism to manage the requirement changes during the distributed development process?
2	Design	The Team members need to evaluate the design of the tool to cover the following aspects: • Look and feel, • Navigation, • Relation between requirements.
3	Usability	How easy to use the tool in different projects?
4	Capability	Does the tool have the required capabilities to support the development steps in the two development phases?
5	Security	Is the tool secure for multi-user environment?
6	Traceability	Can the tool trace the project status?
7	Accessibility	Is the tool accessible from different geographical areas?
8	Multi-user	Will the tool support the usage of multi-user in the same time?
9	Configurations	What is the required configuration to install and use the tool?
10	Generality	Can it used in different type and size of software projects?
11	Tasks assignment	Does the tool provide capability to assign development takes for each task in the development process?

5) Test

The test phase assures that the features are already designed and implemented correctly. This step starts after all developers completing the process of writing codes. Test results are evaluated and sent back to re-implementation in case of test failure.

6) Advance

In this step the customers are involved with development team to evaluate the developed features. The development team can subsequently review the codes and edit it if needed based on the customers' comments. Fig. 3 shows the phases of the suggested approach.

5. The supporting tool

To help in applying the suggested approach a supporting tool is designed. It concentrates on managing the requirements changes in distributed agile development. The tool takes into consideration the nature of distributed agile such as the different geographical locations of the development teams, communication between development team and knowledge management. It is a web based tool intended to facilitate and manage the requirements modeling and changes in all phases. It starts capturing requirements from the project initiation phase and keeps track of requirements and its changes during all next steps and iterations. The tool also allows the project manager to keep track of the status of features in each

Table 3
Findings.

No.	Criteria	Description
1	Concept	• The concept of feature is a good idea, which provides flexibility to the development team to represent requirements and trace the features at the development steps. • The relations between requirements are represented by the cross tree constraints which maintain the dependency between requirements. • The tool allows the development team to have an overview of the project underdevelopment, with focus on the assigned work. • The tool works as a single controller to manage and control the progress of the development. • Constructing the features tree requires a lot of effort at the beginning of the project, and maintaining the feature tree at the development cycle phase also requires a lot of effort and time as well. The consumed time and effort for constructing and maintaining feature tree is acceptable in the light of managing the requirements changes.
2	Design	• The tool is developed as a web based with simple graphical user interface. • It has a navigation menu to facilitate the movement from page to another. • The requirements represented hierarchal in parental relationship, which gives a reasonable and traceable representation of requirements. • The cross-tree relations between requirements are represented by the implication and exclusion constraints. These constraints allow the maintaining of requirements outside the parental relationship.
3	Usability	• The tool is easy to use in large project, and the navigation between features is easy as well. • Easy in use make the tool more usable.
4	Capability	• The tool provided a clear and simple mechanism for managing requirements in distributed agile development. • The tool is divided into two parts; each part is representing a phase of the development according to the proposed approach.
5	Security	• The tool is secure; each team member has his own access password and username. • The project manager, team leaders and team members have different level of permission to access the tool.
6	Traceability	• The tool provides mechanism to trace all steps at the development phases. • It also traces the completed and uncompleted feature for each iteration.
7	Accessibility	• The tool is a web based application and can be hosted on the internet and accessed from anywhere and anytime. • The tool takes the advantages of the web applications.
8	Multi-user	• The tool supports the access of multi-user at the same time. • The tool could be accessed from anywhere.
9	Configurations	• The tool requires a web server to host the web application; in this case it requires the Microsoft Internet Information Server (IIS). • The tool stores the data into Microsoft SQL server database which must be installed as well.
10	Generality	• The tool used in the development of a Document Management System, which is considered as a database systems with medium size. • The tool approved its ability to manage the requirements changes in the context of distributed agile development.
11	Tasks assignment	• The tool doesn't provide capability to allow the team leaders to assign tasks to the development member. • The team leaders are responsible for entering the development data while the development members are able to view and retrieve these data.

development step. Fig. 4 represents the activity diagram of the supporting tool.

Project manager works closely with different parties of stakeholders and development teams to identify the development members, and then the project manager assigns each member to a team. After that the requirements engineers create the initial form of the feature tree, constraints among features are also defined by them. Project manager is involved again in creating the prioritized feature list, each feature list will be assigned to one team. The distributed teams will be involved to record each finished features from the prioritized list in each development cycle step starting from analysis step. The development teams also maintain and record the requirements changes into the feature tree. The development teams keep recording finished features in all steps and the project manager will be able to monitor and track the finished features at each step with the progress of the project. The final step is the acceptance which conducted closely with the customer. Fig. 5 demonstrates how the supporting tool can maintain feature tree.

6. Evaluating the supporting tool

To evaluate this work, we have contacted several software development organizations in Egypt who might be interested in the area of distributed agile development. One of these organizations is the IT department of the National Research Center (NRC) in Egypt. The main responsibilities of the IT

department are developing, maintaining, replacing, and upgrading the software systems for the NRC. NRC is the largest research center in Egypt which has more than 4000 researchers working in different trends of research. The IT department at The NRC showed an interest for evaluating and validating the supporting tool since there is an increasing need for working in distributed development environment. They were about to develop a Document Management System (DMS), the system goal is to initiate, track, manage and store documents in order to reduce paper work. The development teams are located in different building and branches of the NRC; each team is responsible for covering and satisfying the software needs of certain sectors.

Evaluation criteria

After conducting several meetings with the project manager and the team leaders to identify the evaluation criteria, they agreed for these evaluation criteria: concept, design, usability, capabilities, security, traceability, accessibility, multi-user, configurations, generality and tasks assignment [14,15]. Table 2 presents the evaluation criteria with more explanation.

Evaluation teams responsibilities

The responsibilities of the evaluation teams are:

- Elicit and collect requirements from different users.
- Identify the feature of requirements, or translate the user requirements into features.
- Build the feature tree.
- Initiate constraints between features.
- Assign features to different development team; each set of features will form an iteration.
- Estimate the time required for each iteration, according to the project manager vision and the complexity of the features.
- Review and maintain requirements in each iteration, and if needed update the features tree.
- Maintain the relation between features at the development cycle phase.
- Keep track with finished feature at each step in the development phases.

Findings

Table 3 introduces the finding of the tool according to development teams' evaluation.

Recommendations

Table 4 presents the recommendations for the tool which will increase the functionality and usability of the tool in future.

Evaluation summary

The concept of feature is a good idea and it could be customized according to the project nature, it provides flexibility to represent requirements. The relations between requirements are represented by the cross tree constraints which maintain the dependency between requirements. The tool proved its ability to manage the changes of requirements in distributed agile development. It is easy to use and requires reasonable configuration for installation. It allows the development team to contact and cooperate in single place; it also provides a clear framework for working in distributed development. The proposed approach has managed the changes of requirements and proved its ability to maintain the consistency of the requirements. The tool needs to extend its functionality to handle some of the project management tasks like task assignment and graphical follow up. The set of presented recommendation will enhance the functionality of the tool and will increase its usability.

Table 4
Recommendations for the supporting tool.

No.	Criteria	Description
1	Concept	• The tool needs to extend its functionality to consider assigning tasks to the development member. • The tool needs to add some reports to measure the performance of the development team.
2	Design	• The development teams prefer to show the web application messages in popup windows rather than a text on the web page with red color. • The development teams asked to add favorite menu, which will contain the frequently visited pages of the web application.
3	Usability	• The tool won't be usable without the knowledge of the approach. • The tool available only on English and it will be more usable if it supported more languages.
4	Capabilities	• Extend the functionality of the tool to consider assigning tasks to the development member. • The tool should notify the project manager in case of the changes in the feature tree.
5	Security	• The tool should notify the users with their security levels.
6	Tractability	• The tool doesn't have graphical figures to represent the progress of the project.
7	Generality	• The tool approved its ability to manage the requirements changes in the context of distributed agile development for database applications.
8	Tasks assignment	• The tool doesn't provide capability to allow the team leaders to assign takes to the development member.

7. Conclusion and future work

This paper studied different approaches and practices for GAD. The authors discovered that there is a lack for managing requirements changes practices and approaches in GAD. The majority of the existing research in the literature, are in the form industrial experience reports. This paper has combined the feature model with an industrial approach in order to provide a solution that can manage requirements in GAD. The authors have modified the feature model to handle the requirements changes process. The modified feature model used to model requirements for single system and manage its changes, while the original feature model is used mainly in modeling a family of systems or software product line. The modified feature model is called here a features tree. The features tree is integrated with NAPDP which is an approach for distributed agile development. NAPDP consists of two phases, project initiation phase and development cycle phase. Managing requirements changes take place in the two phases, it starts with a project initiation phase and continue through a development cycle phase. This approach is associated with a supporting tool to aid its theoretical vision. The tool is a web based tool to support the distributed agile development. The tool helps to manage the requirements changes and helps the project manager to keep track of project status at the two phases. The supporting tool is evaluated in a real environment at NRC. The project used in the evaluation was a document management system which is medium size database system. The findings of the evaluation are encouraging since they are based on exhaustive set of criteria developed by professional software developers.

In the future, the authors are going to use the tool in different projects to enhance its performance and to provide it with more functionalities. Applying the tool evaluation recommendations will enhance the usability of the tool, and will make it able to be used in many software projects with different sizes and domains.

References

[1] Gorschek T, Wohlin C. Requirements abstraction model. Requir Eng 2006;11:79−101.
[2] Allen Huckabee W. Requirements engineering in an agile software development environment. Defence Acquisition University; 2015.
[3] Paetsch F, Eberlein A, Maurer F. Requirements engineering and agile software development. 2003. p. 308. in null.
[4] Creps D, Klingler C, Levine L, Allemang D. Organization domain modeling (ODM) guidebook version 2.0. Software Technology for Adaptable, Reliable Systems (STARS); 1996.
[5] White Jules, Galindo José A, Saxena Tripti, Dougherty Brian, Benavides David, Schmidt Douglas C. Evolving feature model configurations in software product lines. J Syst Softw January 2014;87:119−36.
[6] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-oriented domain analysis (FODA) feasibility study. 1990. DTIC Document.
[7] Bergner Klaus. DoSAM − domain-specific software architecture comparison model. January 2005. http://dx.doi.org/10.1007/11558569_3.
[8] Arefin MMS, Korzun D. Improvement of the new agile process for distributed projects. 2010.
[9] Korkala Mikko. Customer communication in distributed agile software development. VTT Sci 2015;80.
[10] Henrik K. Scrum and XP from the trenches (Enterprise software development). 2nd ed. 2015. Lulu.com.
[11] Beck K. Extreme programming explained: embrace change. 2nd ed. Addison-Wesley Professional; 2005.
[12] Kruchten P. Rational unified process best practices for software development teams. Canada: Rational Software; 2001.
[13] Borges Pedro, Monteiro Paula, Machado Ricardo J. Mapping RUP Roles to small software development teams. In: SWQD 2012, LNBIP vol. 94; 2012. p. 59−70.
[14] Garcia Jorge Esparteiro, Paiva AnaCR. A requirements-to-implementation mapping tool for requirement traceability. JSW 2015;11(2):193−200.
[15] Embarcadero. Selecting the right change management solution. Embarcadero Technology Technical Notes; 2007.